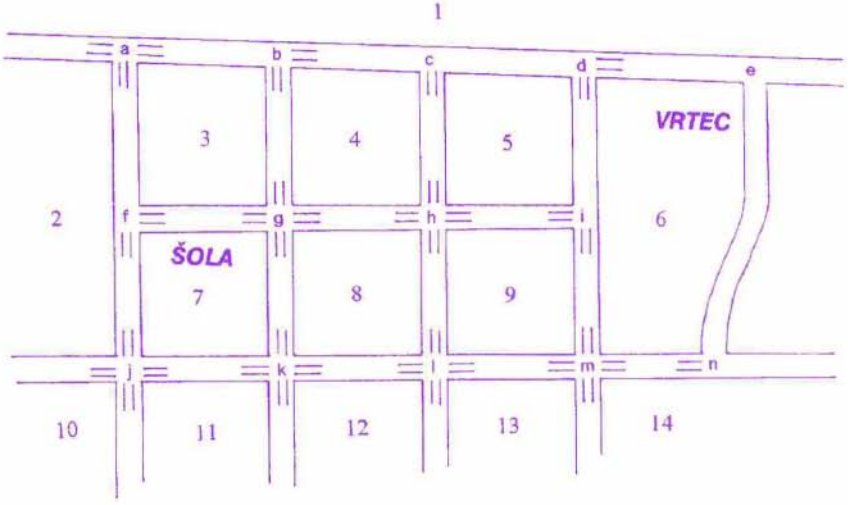
VARNE POTI IN KRATKE POTI

Konec pomladi so se mali šolarji podali na pot v pravo šolo, da si ogledajo, v kaj se bodo podali jeseni. Ker sem z enim izmed udeležencev "pohoda" v tesnem sorodu, pot pa poteka po središču mesta, si nisem mogel kaj, da si ne bi zastavil nekaj vprašanj: Katera pot je najbolj varna? Katera pot je najkrajša?

Za omenjeni primer je shematični zemljevid mestnega okoliša (Maribora) podan na sliki 1, kjer je, kot že rečeno, potrebno priti iz vrtca do šole. Poleg ulic, vrtca in šole so označeni tudi prehodi za pešce, križišča in območja, na katera ceste razdelijo mestno četrt.



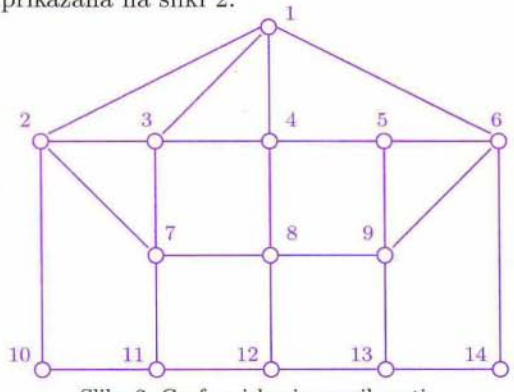
Slika 1. Zemljevid mestne četrti.

Varne poti

Varna pot mora seveda potekati le po pločnikih in prehodih za pešce. Domenimo se, da je najvarnejša tista pot, ki najmanjkrat prečka cesto. Število prečkanj ceste imenujmo *nevarnost* poti. Metoda "ostrega očesa" nas hitro prepriča, da je na sliki 1 nevarnost poti iz vrtca do šole enaka 3 – z manj kot tremi prečkanji ceste žal ne moremo iz vrtca do šole.

Lotimo se sedaj problema v splošnem: Kako bi za poljuben zemljevid in za poljubna začetek in cilj poiskali najvarnejšo pot? Najprej opazimo, da za reševanje problema ne potrebujemo informacije o obliki posameznih območij zemljevida, saj se znotraj območja gibljemo varno. Zato bomo

vsako območje nadomestili z majhnim krožcem. Seveda pa moramo vedeti, iz katerega v katero območje lahko pridemo preko prehoda za pešce. To bomo označili s črto med ustreznima krogcema. Če je med dvema območjema več prehodov, to na varne poti ne vpliva, zato bo ena črta med območjema zadoščala. Za zemljevid s slike 1 dobimo poenostavljeno situacijo, ki je prikazana na sliki 2.

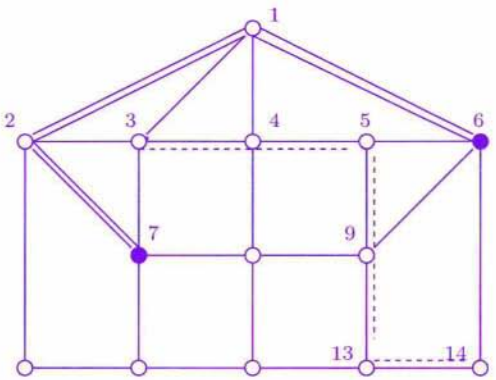


Slika 2. Graf za iskanje varnih poti.

Dobljeno matematično strukturo imenujemo *graf*. O grafih je več zapisano v knjižici iz Presekove knjižnice: *Bajc, Pisanski: Najnujnejše o grafih*. Tu pa povejmo le, da krogcem pravimo *točke*, črtam pa *povezave* grafa. Območju vrtca ustreza točka 6, šolskemu območju pa točka 7.

Po grafu se lahko sprehajamo. To naredimo tako, da gremo iz ene točke v drugo točko po povezavi, ki ju povezuje. Zaporedju takih prehodov rečemo *sprehod* v grafu. Sprehod, v katerem so vse točke različne, imenujemo *pot*. Sprehod med dvema točkama grafa je *najkrajši sprehod*, če vsebuje najmanjše možno število povezav. To število povezav imenujemo *razdalja* med danima točkama. Najkrajši sprehod med dvema točkama je vedno pot. Res, če se na sprehodu ista točka pojavi dvakrat, lahko tisti del sprehoda, ki poteka vmes, opustimo in dobimo krajši sprehod. Zato bomo v nadaljevanju namesto o najkrajših sprehodih govorili kar o najkrajših poteh. Opozorimo še, da lahko med dvema točkama obstaja več različnih najkrajših poti.

Na sliki 3 sta shematično prikazani dve poti v našem grafu. Prva poteka po točkah 6, 1, 2 ter 7 in je ena od treh najkrajših poti med poudarjenima točkama 6 in 7, torej med vrtcem in šolo. Pot po točkah 3, 4, 5, 9, 13 in 14, ki je dolžine 5, pa ni najkrajša pot med 3 in 14, saj lahko med njima hitro najdemo pot dolžine 3.

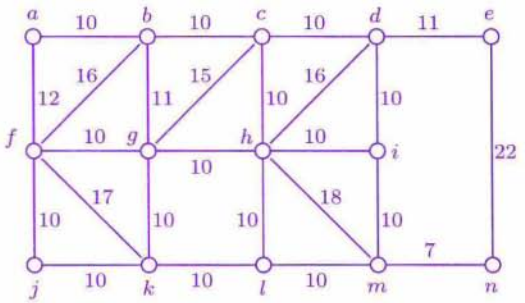


Slika 3. Dve poti v grafu za iskanje varnih poti.

Prečkanje ceste na sliki 1 torej ponazorimo z ustrezno povezavo grafa s slike 2. Zato je nevarnost poti med dvema območjema s slike 1 enaka razdalji med ustreznima točkama s slike 2. Kako v poljubnem grafu poiščemo razdaljo med dvema točkama, bomo povedali nekoliko kasneje, najprej pa si oglejmo naslednji sorodni problem.

Kratke poti

Tokrat bi radi iz vrtca do šole prišli po najkrajši poti. Seveda smemo tudi sedaj prečkati ceste le po prehodih za pešce, ni pa pomembno, kolikokrat prečkamo cesto. Iz zemljevida s slike 1 bomo naredili graf takole. Njegove točke bodo križišča, saj moramo priti v križišče, če želimo prečkati cesto. (Če bi imeli prehode za pešce tudi izven križišč, bi pač tudi tam dodali točke.) Nato dve točki povežemo, če gre za sosednji križišči. Seveda pa potrebujemo tudi dodatno informacijo – dejanske razdalje med posameznimi točkami. Tako dobimo graf, ki je prikazan na sliki 4. Tak graf imenujemo *uteženi graf*. Vrednostim na povezavah bomo rekli *cene*, saj lahko te vrednosti predstavljajo tudi kaj drugega kot dolžino. Na primer znesek, ki ga moramo plačati za cestnino, ali pa ceno izgradnje posameznega odseka. Za naš primer si lahko predstavljamo, da so cene povezav desetkratniki dejanskih vrednosti, tako na primer cena 10 predstavlja razdaljo 100 metrov.



Slika 4. Graf za iskanje najcenejših poti.

Opozorimo, da so s povezavo povezana tudi nekatera križišča, ki jih sprva morda ne bi povezali, na primer križišči h in m. Pomen takih povezav je v "bližnjicah", ki potekajo znotraj posameznih območij, v tem primeru po območju številka 9. Zopet drugje nimamo diagonal, na primer med križiščema b in h, saj ni bližnjice med njima znotraj območja 4. Skratka, v primeru iskanja najkrajših poti je pomembno tudi poznavanje notranjosti območij.

Vrednost poti v uteženem grafu je definirana kot vsota cen vseh povezav na poti. Najkrajša pot je seveda tista z najmanjšo možno vrednostjo. Na primer, vrednost poti $f - k - l - h - d$ je $17 + 10 + 10 + 16 = 53$. To ni najkrajša pot med točkama f in d, saj je vrednost poti $f - a - b - c - d$ enaka 42. Je to najkrajša pot med f in d?

Če na vsako povezavo grafa zapišemo vrednost 1, je vrednost najkrajše poti med dvema točkama enaka nevarnosti med njima. Seveda moramo paziti, v katerem grafu to naredimo: za varne poti moramo to narediti na grafu s slike 2. Če torej imamo postopek, ki v uteženem grafu poišče vrednosti najkrajših poti, z njim lahko poiščemo tudi nevarnosti med točkami.

Iskanje vrednosti najkrajših poti

Sedaj bomo opisali algoritem, ki za vsak par točk uteženega grafa poišče vrednost najkrajše poti med njima. Kot bomo videli, je postopek zelo enostaven. Metode, s katero pridemo do algoritma, ne bomo razložili (mimogrede, imenuje se dinamično programiranje), prav tako pa ne bomo strogo dokazali, da algoritem deluje pravilno. Oboje namreč presega okvir tega prispevka.

Algoritem je takle. Recimo, da ima obravnavani graf n točk. Njegove točke oštevilčimo s števili od 1 do n. Nato v dvorazsežno polje, imenujmo ga `razdalje`, zapišemo cene posameznih povezav. Natančneje, če sta točki i in j povezani s povezavo, potem v `razdalje[i,j]` zapišemo ceno povezave med i in j, sicer pa zapišemo ∞. V računalniku seveda ne moremo zapisati vrednosti neskončno, zato zapišemo dovolj veliko število, na primer 100-krat večje od tipičnih cen povezav. V spodnjem programu smo izbrali vrednost 99999. No, priročneje je, če za točki, ki nista povezani s povezavo, kot podatek vnesemo vrednost -1, to pa potem s programom spremenimo v 99999. S tem smo pripravili podatke za jedro algoritma.

Ko so podatki pripravljeni oziroma prebrani, se sprehodimo po vseh točkah. Za vsako točko, imenujmo jo i, se vprašamo, ali morda ne poteka bližnjica iz neke točke j do neke druge točke k preko točke i. Pri izbrani točki i to preverimo za vse možne pare točk. Če taka bližnjica obstaja, si to seveda zapomnimo. Ko se algoritem izteče, imamo v polju `razdalje` zapisane vrednosti najkrajših poti za vse pare točk uteženega grafa. Zapišimo ta algoritem v obliki programa.

```
program najcenejse_poti;
{ V uteženem grafu za vsak par točk poišče vrednost najkrajše poti }
{ med njima. }

const
  maximum = 100;                { Največje dovoljeno število točk grafa. }
var
  razdalje: array [1..maximum,1..maximum] of longint;
  i, j, k, n: integer;
  ime: string[20];              { Ime datoteke, na kateri so podatki o cenah. }
  f: text;
begin
  { Pripravimo datoteko, na kateri so vpisani podatki. }
  write('Vhodna datoteka: '); readln(ime);
  assign(f,ime); reset(f);
  { Z datoteke f preberemo število točk in cene vseh povezav. }
  readln(f,n);
  for i := 1 to n do begin
    for j := 1 to n do begin
      read(f,razdalje[i,j]);
      if razdalje[i,j]=-1 then razdalje[i,j] := 99999;
    end;
    readln(f);
  end;
```

```
  { Bistvo programa so naslednji trije stavki for. }
  { Vprašamo se: ali imamo bližnjico iz j v k preko i. }
  for i := 1 to n do
    for j := 1 to n do
      for k := 1 to n do
    if razdalje[j,k] > razdalje[j,i] + razdalje[i,k] then
    razdalje[j,k] := razdalje[j,i] + razdalje[i,k];
  { Samo še izpis rezultatov, pa smo končali. }
  for i := 1 to n do begin
    for j := 1 to n do write(razdalje[i,j]:3);
    writeln;
  end;
end.
```

Seveda bi lahko gornji postopek sprogramirali tudi elegantneje, vendar bi s tem zameglili njegovo bistvo. Na primer, podatke bi lahko pripravili (in brali) v taki obliki, da bi vnesli samo neničelne cene, dovolj pa bi tudi bilo vnesti le polovico podatkov, saj vedno velja `razdalje[i,j] = razdalje[j,i]`.

Omenimo še, da opisani postopek ni edini, ki poišče vrednosti najkrajših poti. Ima tudi pomanjkljivost, saj kot rezultat dobimo le vrednosti najkrajših poti, ne pa tudi samih poti. Seveda pa smo s tremi preprostimi stavki `for` dobili več, kot bi si lahko mislili na začetku.

Primera

Preizkusimo gornji program na obeh primerih iz tega prispevka. Graf s slike 2 ima 14 točk, vhodni podatki, ki jih pripravimo na datoteki, pa so naslednji:

```
14
 0  1  1  1 -1  1 -1 -1 -1 -1 -1 -1 -1 -1
 1  0  1 -1 -1 -1  1 -1 -1  1 -1 -1 -1 -1
 1  1  0  1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1
 1 -1  1  0  1 -1 -1  1 -1 -1 -1 -1 -1 -1
-1 -1 -1  1  0  1 -1 -1  1 -1 -1 -1 -1 -1
 1 -1 -1 -1  1  0 -1 -1  1 -1 -1 -1 -1  1
-1  1  1 -1 -1 -1  0  1 -1 -1  1 -1 -1 -1
-1 -1 -1  1 -1 -1  1  0  1 -1 -1  1 -1 -1
-1 -1 -1 -1  1  1 -1  1  0 -1 -1 -1  1 -1
-1  1 -1 -1 -1 -1 -1 -1 -1  0  1 -1 -1 -1
-1 -1 -1 -1 -1 -1  1 -1 -1  1  0  1 -1 -1
-1 -1 -1 -1 -1 -1 -1  1 -1 -1  1  0  1 -1
-1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1  1  0  1
-1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1  1  0
```

Izpis, ki ga vrne program, je prikazan v spodnji tabeli:

```
0  1  1  1  2  1  2  2  2  2  3  3  3  2
1  0  1  2  3  2  1  2  3  1  2  3  4  3
1  1  0  1  2  2  1  2  3  2  2  3  4  3
1  2  1  0  1  2  2  1  2  3  3  2  3  3
2  3  2  1  0  1  3  2  1  4  4  3  2  2
1  2  2  2  1  0  3  2  1  3  4  3  2  1
2  1  1  2  3  3  0  1  2  2  1  2  3  4
2  2  2  1  2  2  1  0  1  3  2  1  2  3
2  3  3  2  1  1  2  1  0  4  3  2  1  2
2  1  2  3  4  3  2  3  4  0  1  2  3  4
3  2  2  3  4  4  1  2  3  1  0  1  2  3
3  3  3  2  3  3  2  1  2  2  1  0  1  2
3  4  4  3  2  2  3  2  1  3  2  1  0  1
2  3  3  3  2  1  4  3  2  4  3  2  1  0
```

Najnevarnejše poti so tiste med 2 in 13, 3 in 13, 5 in 10, 5 in 11, 6 in 11, 7 in 14, 9 in 10 ter 10 in 14, pri vseh moramo (vsaj) štirikrat prečkati cesto.

V drugem primeru pa preizkusimo program na uteženem grafu s slike 4. Tudi ta graf ima 14 točk, ki jih oštevilčimo na naraven način, torej a naj bo 1 in n naj bo 14. Podatki za program so:

```
14
 0 10 -1 -1 -1 12 -1 -1 -1 -1 -1 -1 -1 -1
10  0 10 -1 -1 16 11 -1 -1 -1 -1 -1 -1 -1
-1 10  0 10 -1 -1 15 10 -1 -1 -1 -1 -1 -1
-1 -1 10  0 11 -1 -1 16 10 -1 -1 -1 -1 -1
-1 -1 -1 11  0 -1 -1 -1 -1 -1 -1 -1 -1 22
12 16 -1 -1 -1  0 10 -1 -1 10 17 -1 -1 -1
-1 11 15 -1 -1 10  0 10 -1 -1 10 -1 -1 -1
-1 -1 10 16 -1 -1 10  0 10 -1 -1 10 18 -1
-1 -1 -1 10 -1 -1 -1 10  0 -1 -1 -1 10 -1
-1 -1 -1 -1 -1 10 -1 -1 -1  0 10 -1 -1 -1
-1 -1 -1 -1 -1 17 10 -1 -1 10  0 10 -1 -1
-1 -1 -1 -1 -1 -1 -1 10 -1 -1 10  0 10 -1
-1 -1 -1 -1 -1 -1 -1 18 10 -1 -1 10  0  7
-1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 -1  7  0
```

In še rezultat programa:

```
 0 10 20 30 41 12 21 30 40 22 29 39 48 55
10  0 10 20 31 16 11 20 30 26 21 30 38 45
20 10  0 10 21 25 15 10 20 35 25 20 28 35
30 20 10  0 11 35 25 16 10 45 35 26 20 27
41 31 21 11  0 46 36 27 21 56 46 37 29 22
12 16 25 35 46  0 10 20 30 10 17 27 37 44
21 11 15 25 36 10  0 10 20 20 10 20 28 35
30 20 10 16 27 20 10  0 10 30 20 10 18 25
40 30 20 10 21 30 20 10  0 40 30 20 10 17
22 26 35 45 56 10 20 30 40  0 10 20 30 37
29 21 25 35 46 17 10 20 30 10  0 10 20 27
39 30 20 26 37 27 20 10 20 20 10  0 10 17
48 38 28 20 29 37 28 18 10 30 20 10  0  7
55 45 35 27 22 44 35 25 17 37 27 17  7  0
```

Najdaljša pot med vsemi najkrajšimi potmi je dolžine 56 in poteka med točkama e in j, druga najdaljša pa je pot med točkama a in n. Poišči ju! Vse ostale poti so krajše od 50.

Naloga

Za svoj okoliš ugotovi dejanske podatke o prometni ureditvi in nato poišči najvarnejše in najkrajše poti. Pri tem lahko varnost poti še nekoliko izpopolniš: če cesto prečkamo pri semaforju, je to varneje, kot če jo prečkamo na nesemaforiziranem križišču (na primer dvakrat varneje). Kako moramo za ta primer pripraviti podatke?

Sandi Klavžar

HITRO KVADRIRANJE ŠTEVIL, KI SE KONČUJEJO S ŠTEVKO 5

Če kvadriramo število, katerega zadnja števka v desetiškem zapisu je enaka 5, dobimo število, ki se končuje s 25. Tudi števke, ki stojijo pred tem dvomestnim koncem, lahko dobimo na zelo preprost način (samo iz števk, ki so pred zaključno števko 5 v začetnem številu).

1. Zapiši nekaj primerov in poskusi iz njih uganiti, po kakšnem pravilu dobimo začetne števke kvadratov takih števil.
2. Zakaj pravilo vedno deluje?
3. Ali lahko še v kakšni bazi, različni od 10, dobimo podobno pravilo za kvadriranje nekaterih števil?

Marija Vencelj